Practical_Unsupervised_Learning_on_Iris_Data_Using_R

Quyen Di Sabino

2025-09-06

```r
knitr::opts_chunk$set(echo = TRUE, warning = FALSE)
```

Introduction

The Iris dataset contains 150 observations of iris flowers from three different species, with four numerical features describing each sample (sepal length, sepal width, petal length, and petal width). While the species labels are available, we treat this dataset as unlabeled to simulate real-world unsupervised learning scenarios.

Using R, we apply a range of unsupervised learning methods, including:

1. Dimensionality Reduction: PCA, t-SNE, and UMAP to project high-dimensional data into 2D space for visualization.
2. Clustering Algorithms: K-Means, Hierarchical Clustering, Density-Based methods DBSCAN to group similar data points.
3. Cluster Evaluation: Silhouette scores, NbClust, and visual methods to assess the quality and structure of clusters.

This project aims to demonstrate how unsupervised learning techniques can be used to explore data, identify meaningful clusters, and reduce dimensionality. All without relying on predefined labels.

Through visualization and analysis, we compare the performance of various clustering techniques and interpret how well they recover the underlying structure of the data.

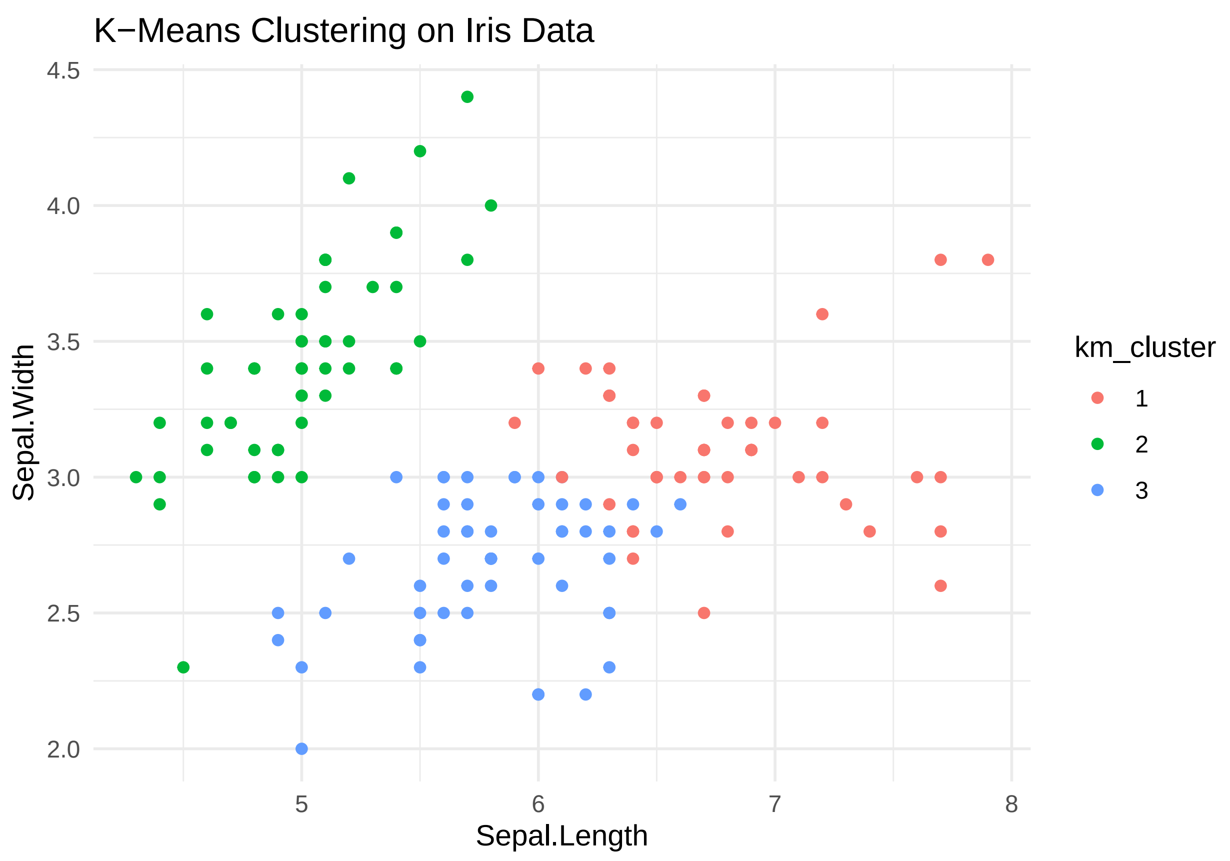
```r
# Load dataset
data("iris")

# Remove the species column (unsupervied = no labels) and scale the data
iris_data <- scale(iris[, 1:4])

# Preview data
head(iris_data)
```

```
##      Sepal.Length Sepal.Width Petal.Length Petal.Width
## [1,]   -0.8976739  1.01560199    -1.335752   -1.311052
## [2,]   -1.1392005 -0.13153881    -1.335752   -1.311052
## [3,]   -1.3807271  0.32731751    -1.392399   -1.311052
## [4,]   -1.5014904  0.09788935    -1.279104   -1.311052
## [5,]   -1.0184372  1.24503015    -1.335752   -1.311052
## [6,]   -0.5353840  1.93331463    -1.165809   -1.048667
```

```r
#------------------------------
# kmean() K-Means Clutering
#------------------------------


# Set seed for reproducibility
set.seed(79)

# Apply K-Means Clustering with 3 clusters
km <- kmeans(iris_data, centers = 3, nstart = 25)

# View cluster assignments
km$cluster
```

```
##   [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [38] 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 3 3 3 1 3 3 3 3 3 3 3 3 1 3 3 3 3 1 3 3 3
##  [75] 3 1 1 1 3 3 3 3 3 3 3 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 1 1 1 1 3 1 1 1 1
## [112] 1 1 3 3 1 1 1 1 3 1 3 1 3 1 1 3 1 1 1 1 1 1 3 3 1 1 1 3 1 1 1 3 1 1 1 3 1
## [149] 1 3
```

```r
# Add cluster to iris dataset
iris$km_cluster <- as.factor(km$cluster)

# Visualize with ggplot2
library(ggplot2)
ggplot(iris, aes(Sepal.Length, Sepal.Width, color = km_cluster)) +
  geom_point() +
  labs(title = "K-Means Clustering on Iris Data") +
  theme_minimal()
```

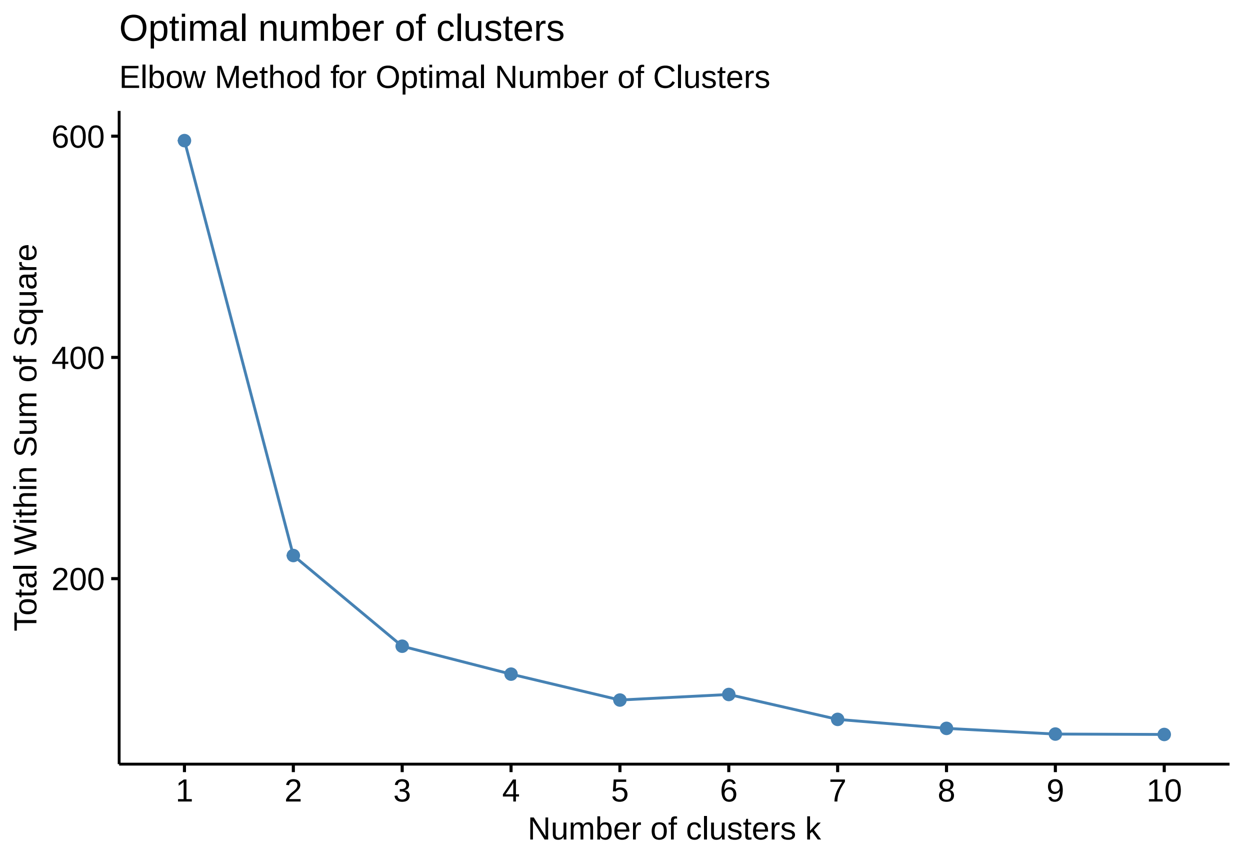


```r
# How to chose the number of clusters (Elbow method)
if(!require(factoextra)) install.packages("factoextra")
```

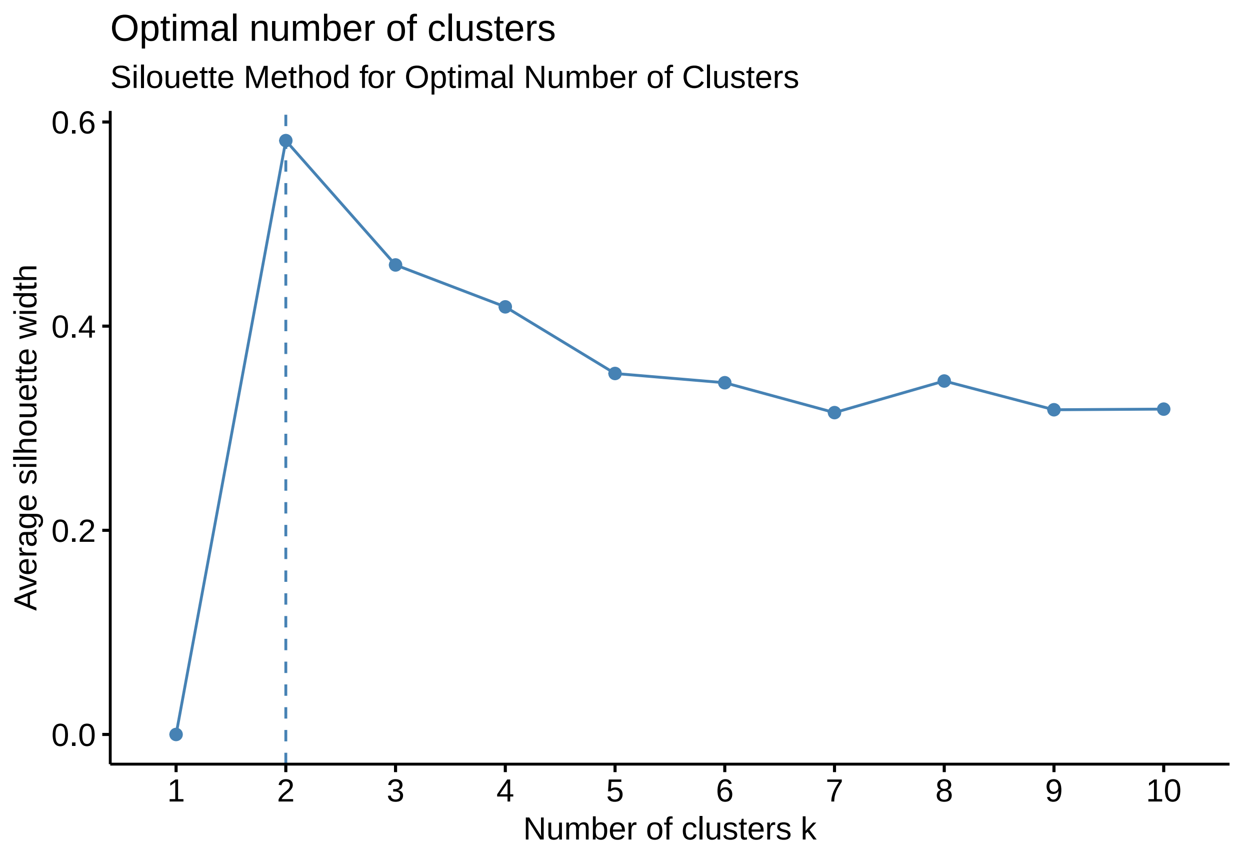
```
## Loading required package: factoextra
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

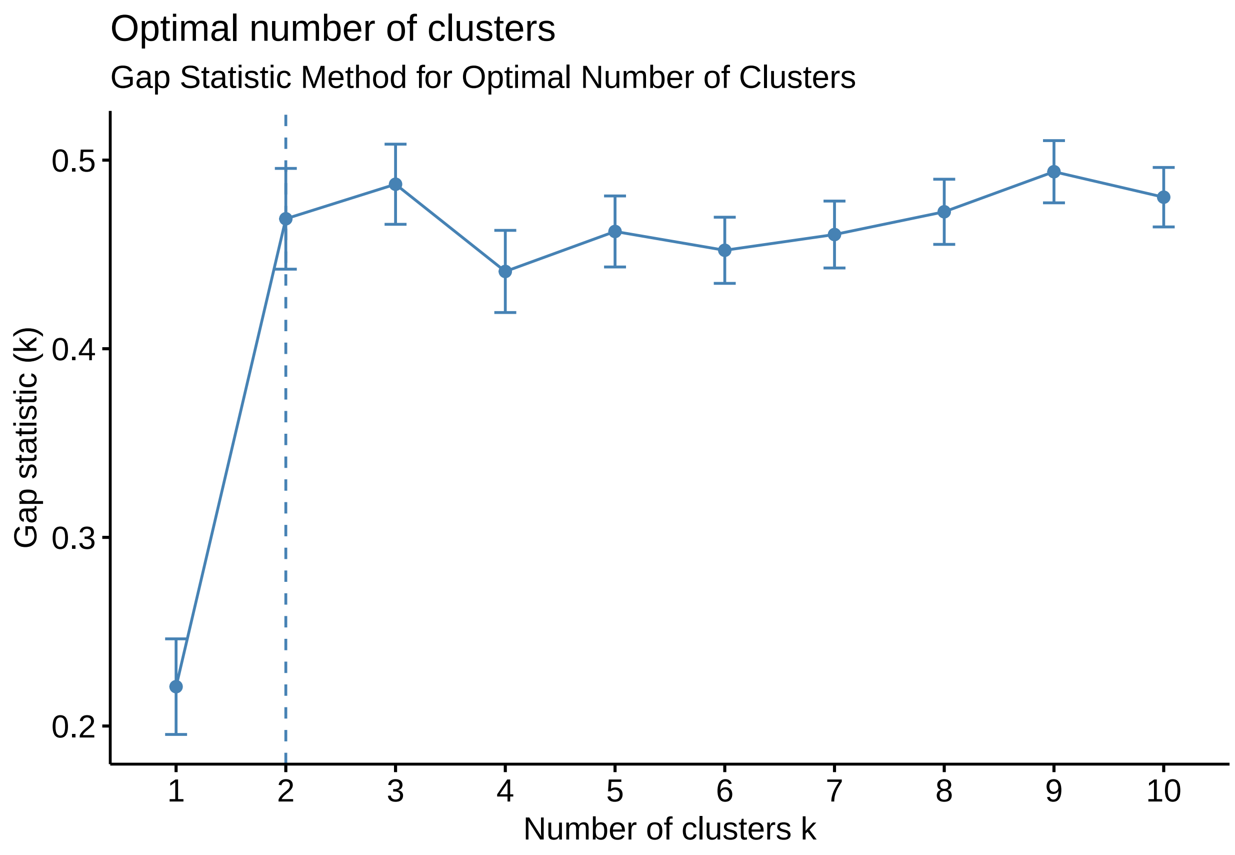
```r
library(factoextra)

# Plot tot.withinss for k = 1 to k = 10 by default
fviz_nbclust(iris_data, kmeans, method = "wss") +
  labs(subtitle = "Elbow Method for Optimal Number of Clusters")
```



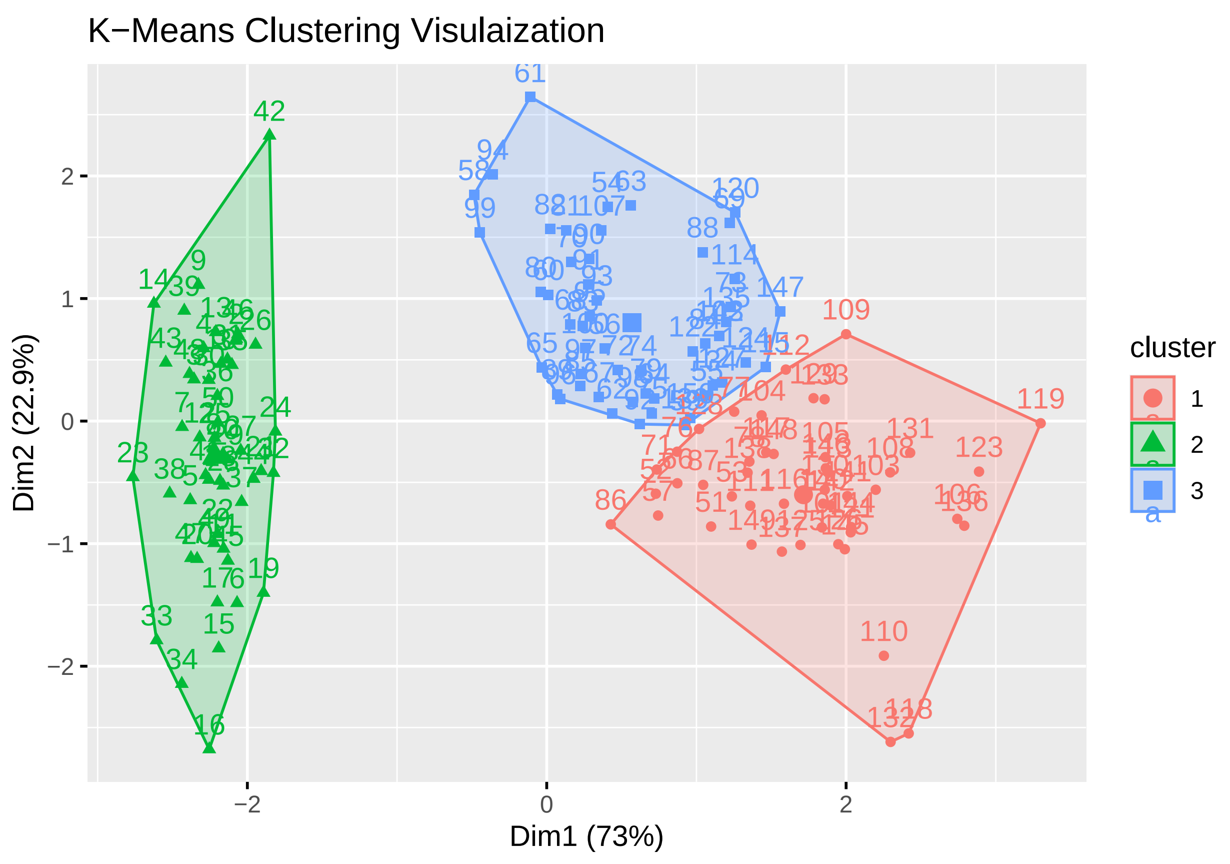
```r
# How to chose the number of clusters (Silhouette Width method)
# Choose the number of clusters that maximizes the average silhouette width
fviz_nbclust(iris_data, kmeans, method = "silhouette") +
  labs(subtitle = "Silouette Method for Optimal Number of Clusters")
```



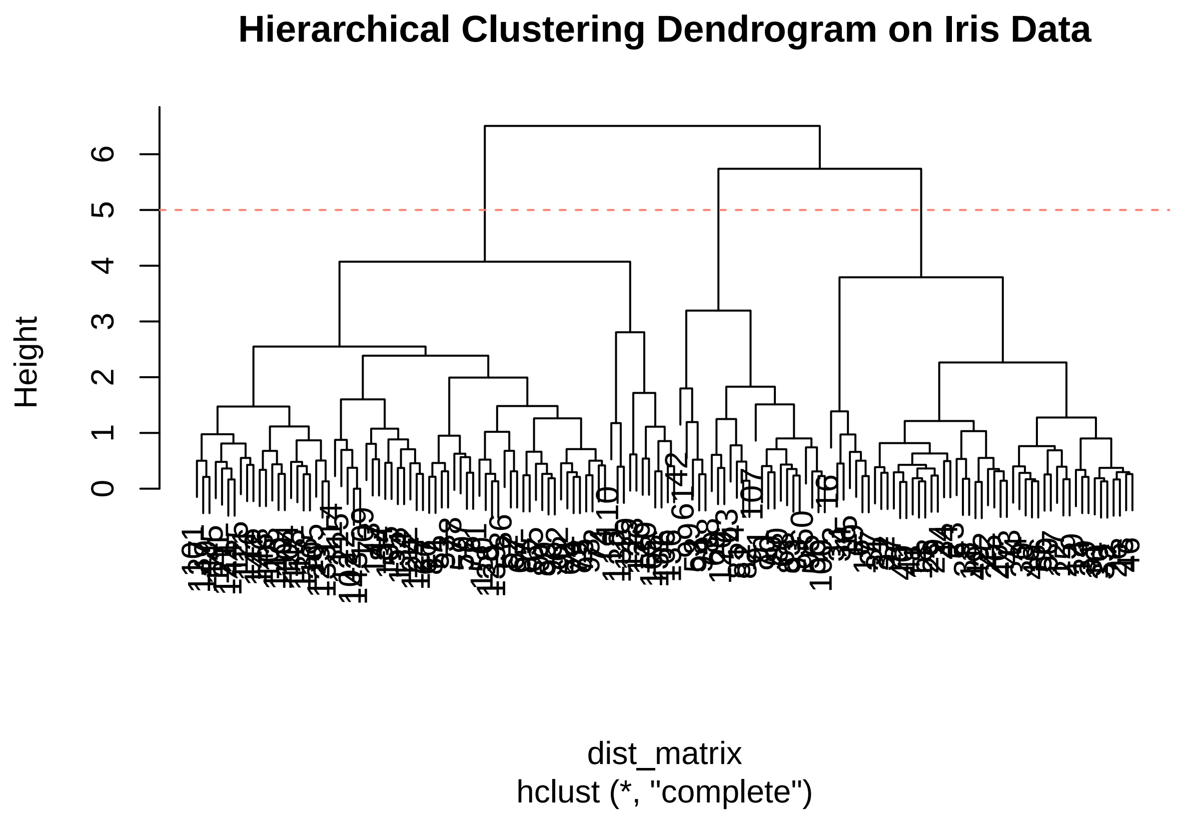
```
# How to chose the number of clusters (Gap Statistic method)
# Look for biggest gap
# Gap statistic calculation (Take a bit longer to run)
set.seed(79)
fviz_nbclust(iris_data, kmeans, method = "gap_stat", nstart = 25, nboot = 50) +
  labs(subtitle = "Gap Statistic Method for Optimal Number of Clusters")
```



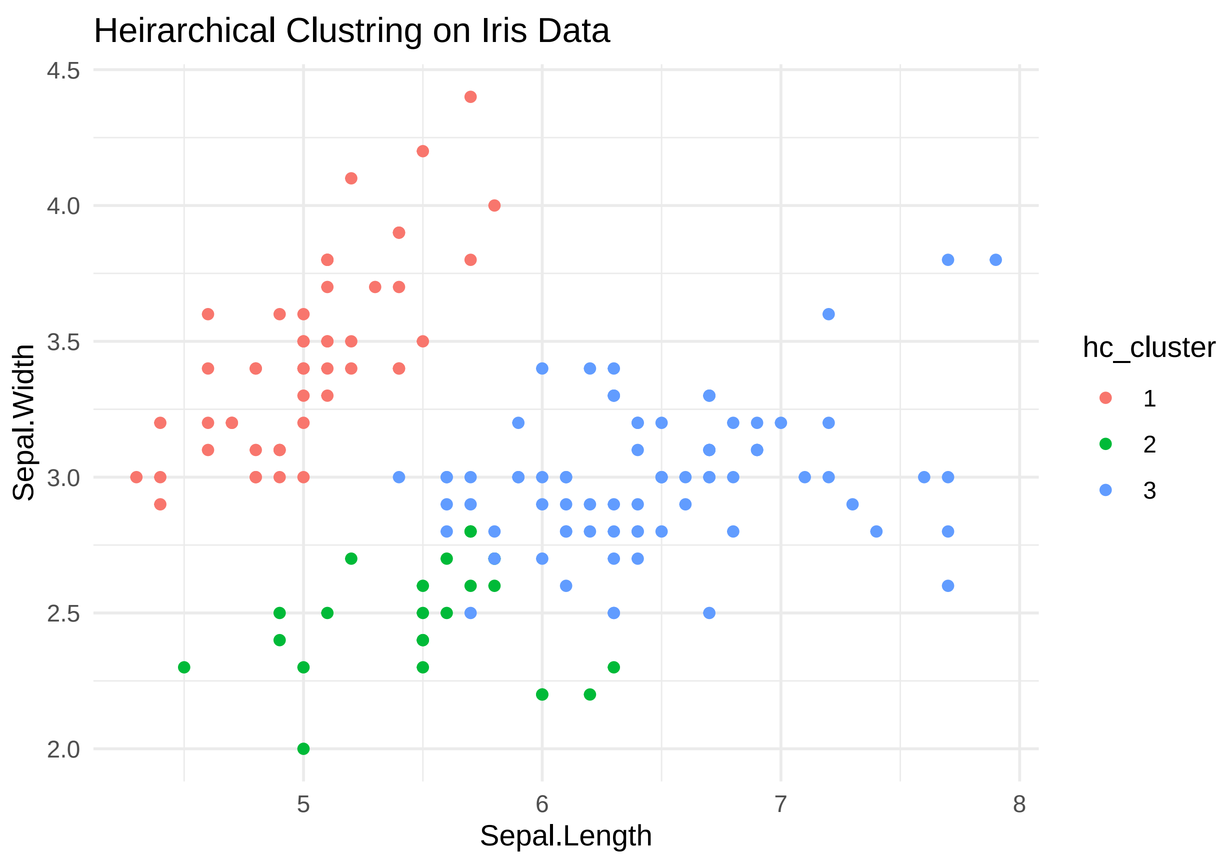
```r
# Visualized clusters using factoextra::fviz_cluster()
# This plot the first two principal components by default and color point by cluster
fviz_cluster(km, data = iris_data) +
  labs(title = "K-Means Clustering Visulaization")
```



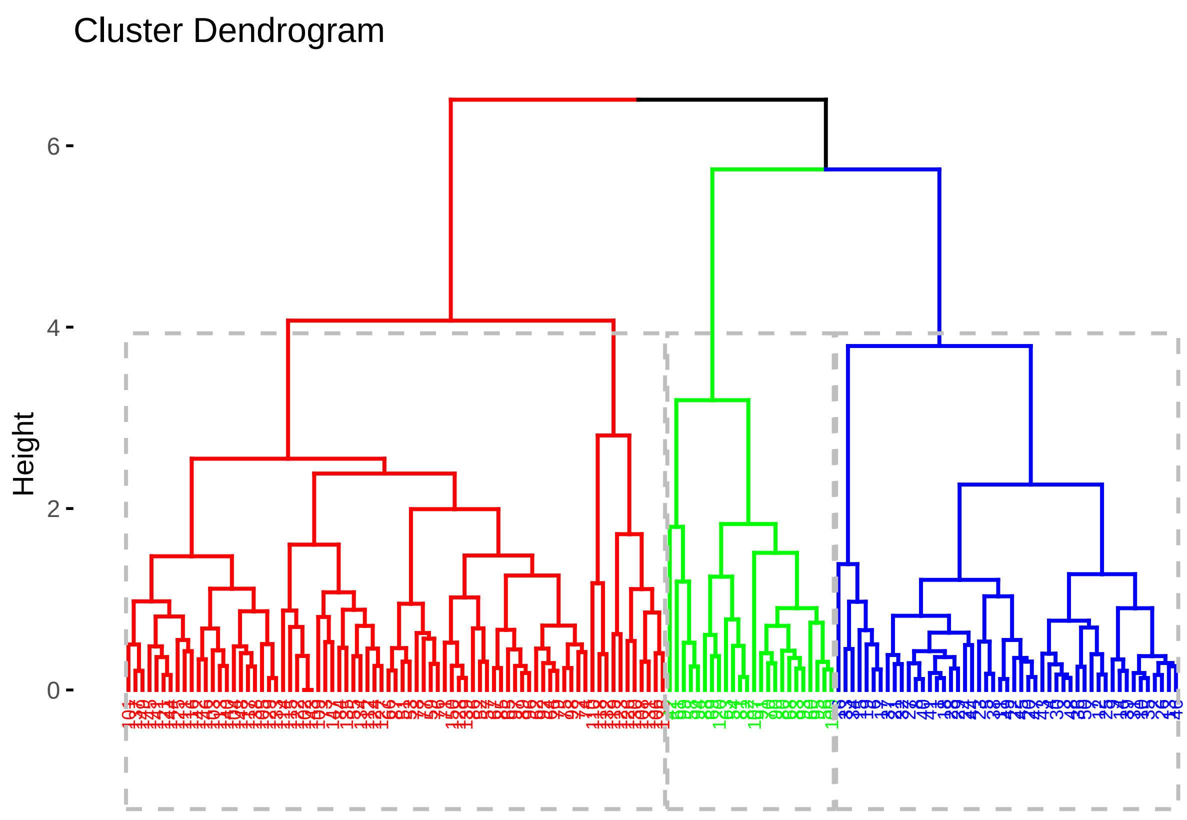
```r
#------------------------------------
# hclust() Hierarchical Clustering
#------------------------------------


# Compute distance metrix
dist_matrix <- dist(iris_data)

# Apply hierarchical clustering
hc <- hclust(dist_matrix)

# Plot dendrogram
plot(hc, main = "Hierarchical Clustering Dendrogram on Iris Data")
abline(h = 5, col = "salmon", lty = 2)
```



```
# Cut the tree into 3 clusters or specify h. cutree(hc, h = 5)
hc_clusters <- cutree(hc, k = 3)

# Add to Iris Dataset
iris$hc_cluster <- as.factor(hc_clusters)

# Plot clusters
ggplot(iris, aes(Sepal.Length, Sepal.Width, color = hc_cluster)) +
  geom_point() +
  labs(title = "Heirarchical Clustring on Iris Data") +
  theme_minimal()
```



```
# Visual dendrogram with clusters highlighted (cut into 3 clusters)
fviz_dend(hc, k = 3,
          cex = 0.5, # label size
          k_colors = c("red", "green", "blue"), # cluster colors
          color_labels_by_k = TRUE,
          rect = TRUE # add rectangles around clusters
          )
```



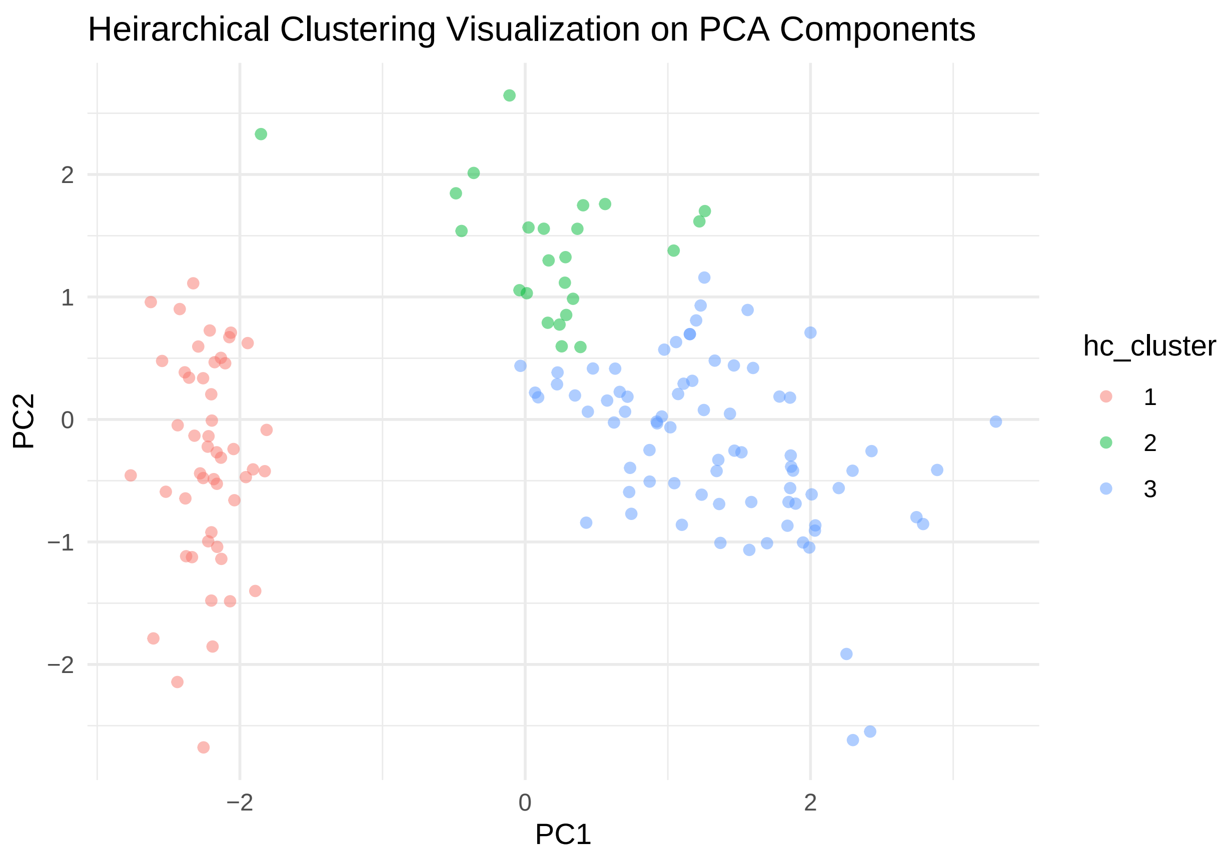
```r
#-------------------------------------------------
# prcomp() Principal Component Analysis (PCA)
#-------------------------------------------------


# Run PCA on standardized data
pca <- prcomp(iris[, 1:4], scale. = TRUE)

# Summary of variance explained
summary(pca)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4
## Standard deviation     1.7084 0.9560 0.38309 0.14393
## Proportion of Variance 0.7296 0.2285 0.03669 0.00518
## Cumulative Proportion  0.7296 0.9581 0.99482 1.00000
```

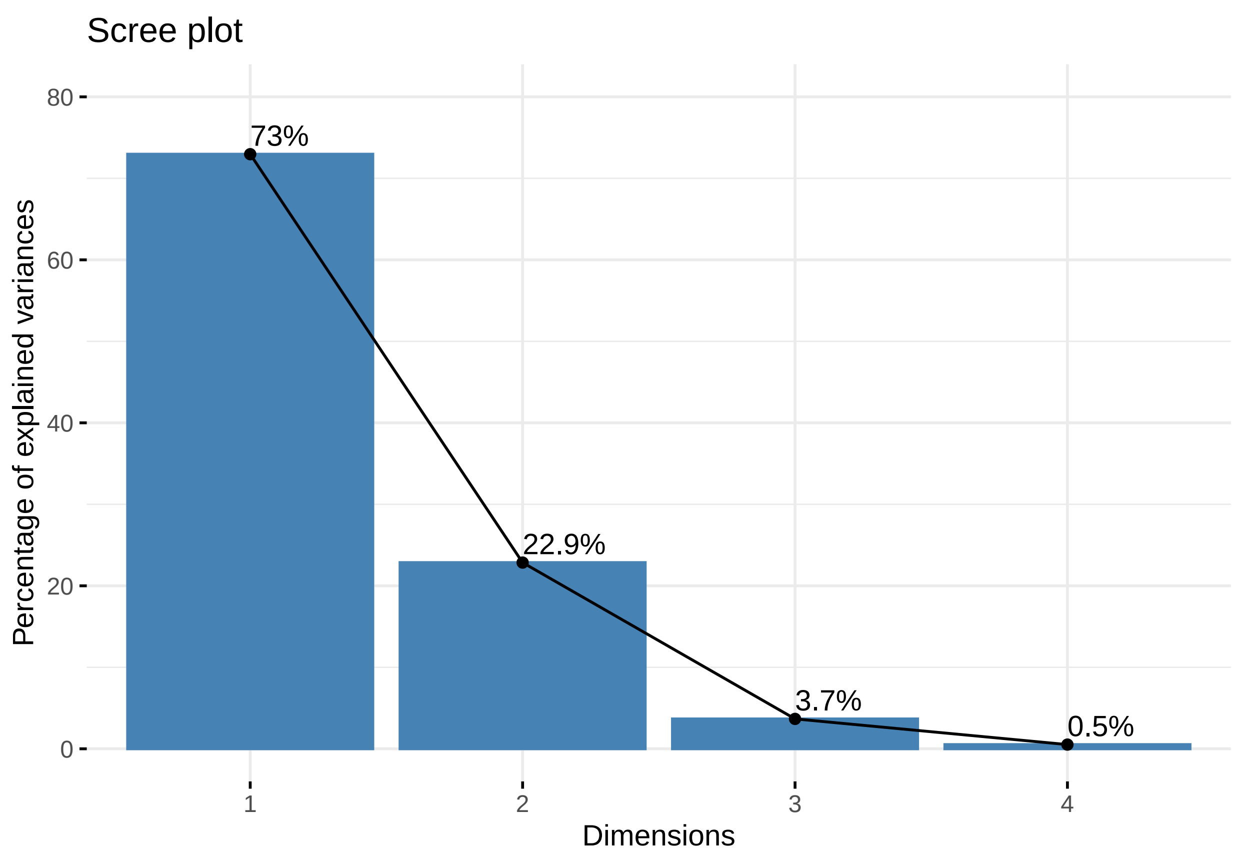
```r
# PCA loading, how original variables contribute to PCs
pca$rotation
```

```
##                     PC1         PC2        PC3        PC4
## Sepal.Length  0.5210659 -0.37741762  0.7195664  0.2612863
## Sepal.Width  -0.2693474 -0.92329566 -0.2443818 -0.1235096
## Petal.Length  0.5804131 -0.02449161 -0.1421264 -0.8014492
## Petal.Width   0.5648565 -0.06694199 -0.6342727  0.5235971
```

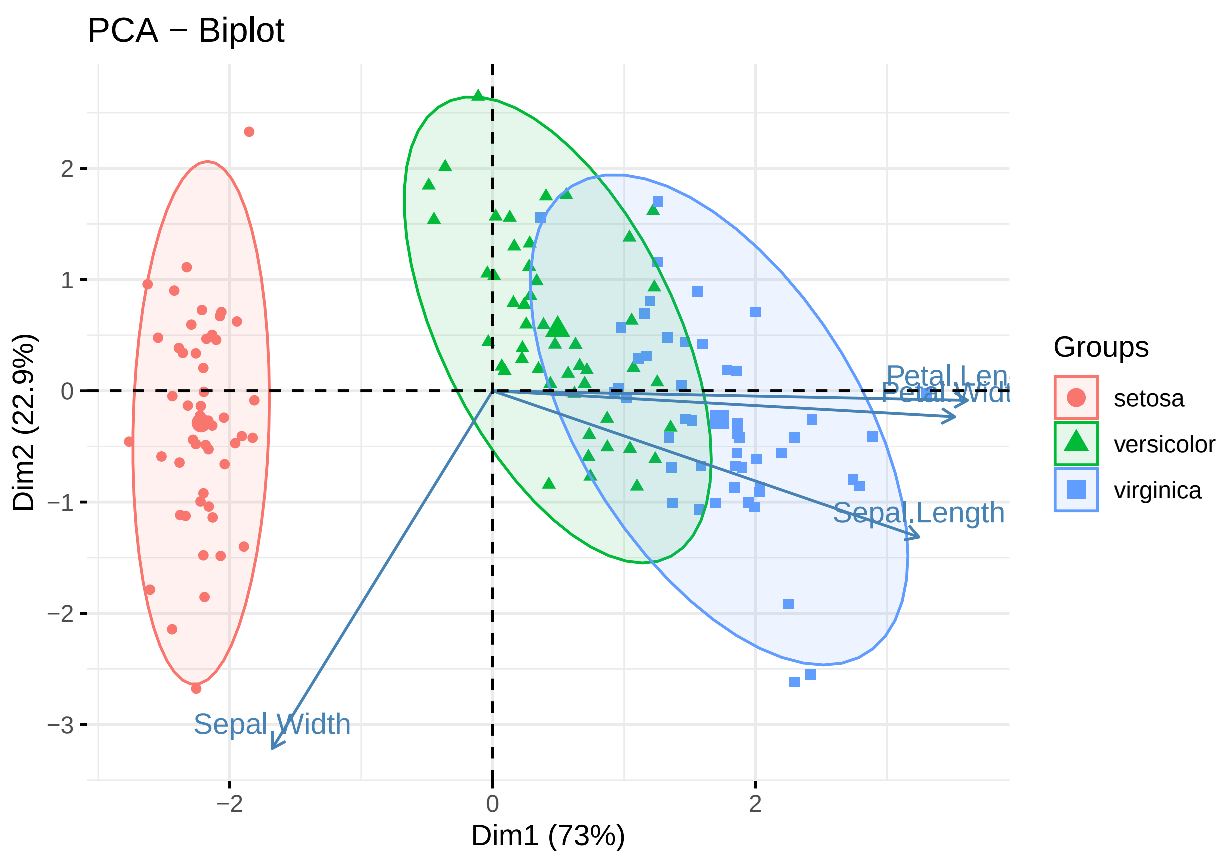
```r
# PCA scores, new coordinates for each data point
iris_pca <- as.data.frame(pca$x)

# Plot PCA components colored by hierarchical clusters
iris_pca$hc_cluster <- as.factor(hc_clusters)
ggplot(iris_pca, aes(PC1, PC2, color = hc_cluster)) +
  geom_point(alpha = 0.5) +
  labs(title = "Heirarchical Clustering Visualization on PCA Components") +
  theme_minimal()
```



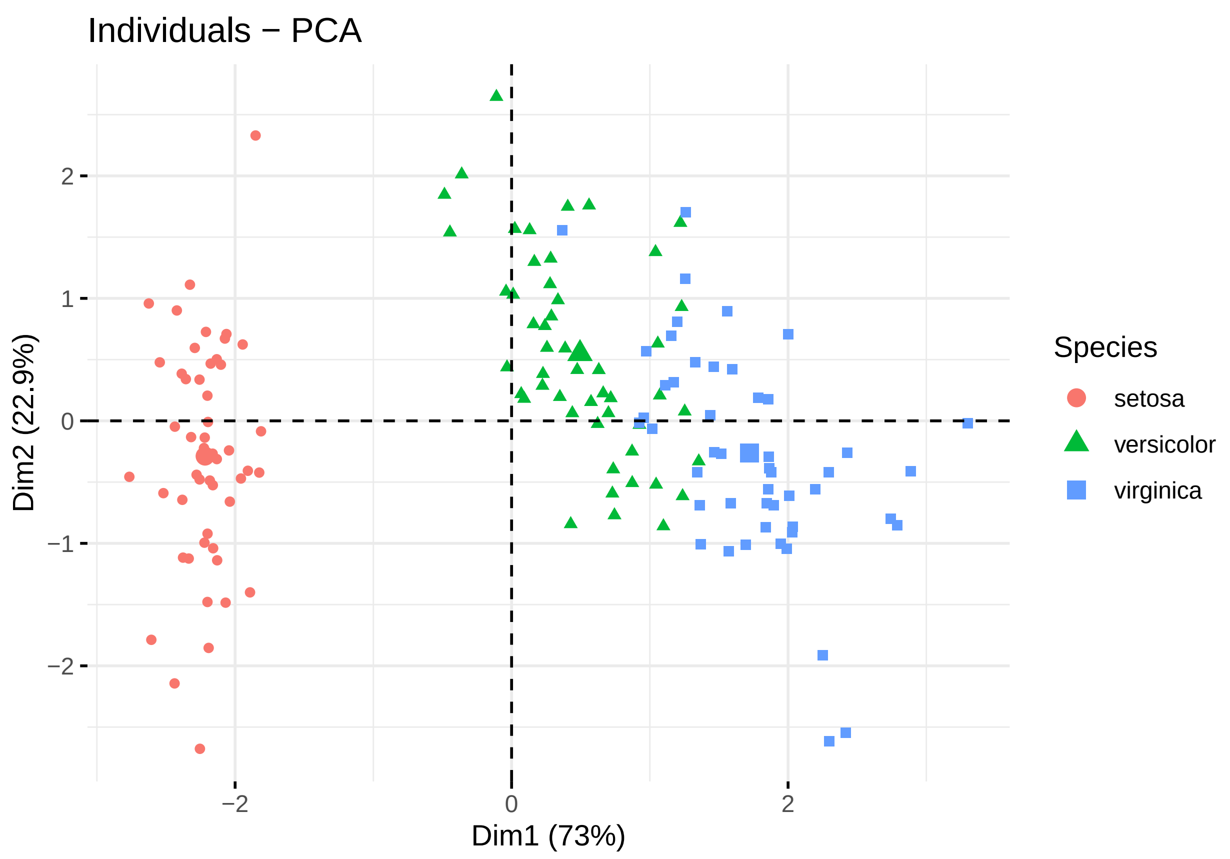
```r
# Scree plot, show how much variance each PC explains
fviz_eig(pca, addlabels = TRUE, ylim = c(0, 80))
```



```
# Biplot, scores and loading in one plot (observations and original variables)
# Arrows show variable contributions
fviz_pca_biplot(pca,
                label = "var", # Only label the variables
                habillage = iris$Species, # Color points by species
                addEllipses = TRUE)
```



```
# Or PCA plot (2D projection of data) or using ggplot2
# Personally, I prefer biplot
fviz_pca_ind(pca,
             geom.ind = "point",
             col.ind = iris$Species,
             palette = TRUE, # use default color palette
             legend.title = "Species")
```

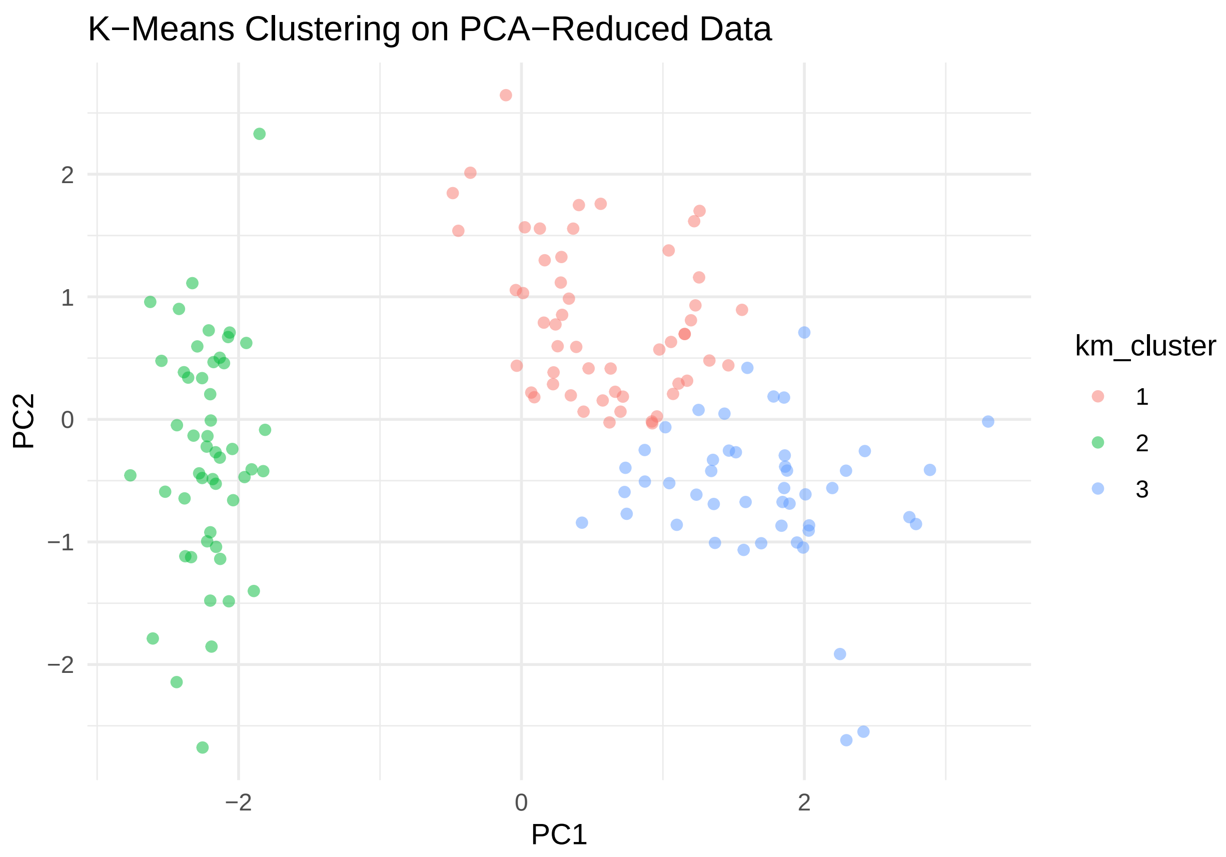


```r
#--------------------------------
# dist() Distance Calculation
#--------------------------------


# Calculate Euclidean distance between observations
dist_matrix <- dist(iris_data)

# Convert to matrix to review
distance_matrix_matrix <- as.matrix(dist_matrix)

# View top left corner of matrix
round(distance_matrix_matrix[1:5, 1:5], 2)
```

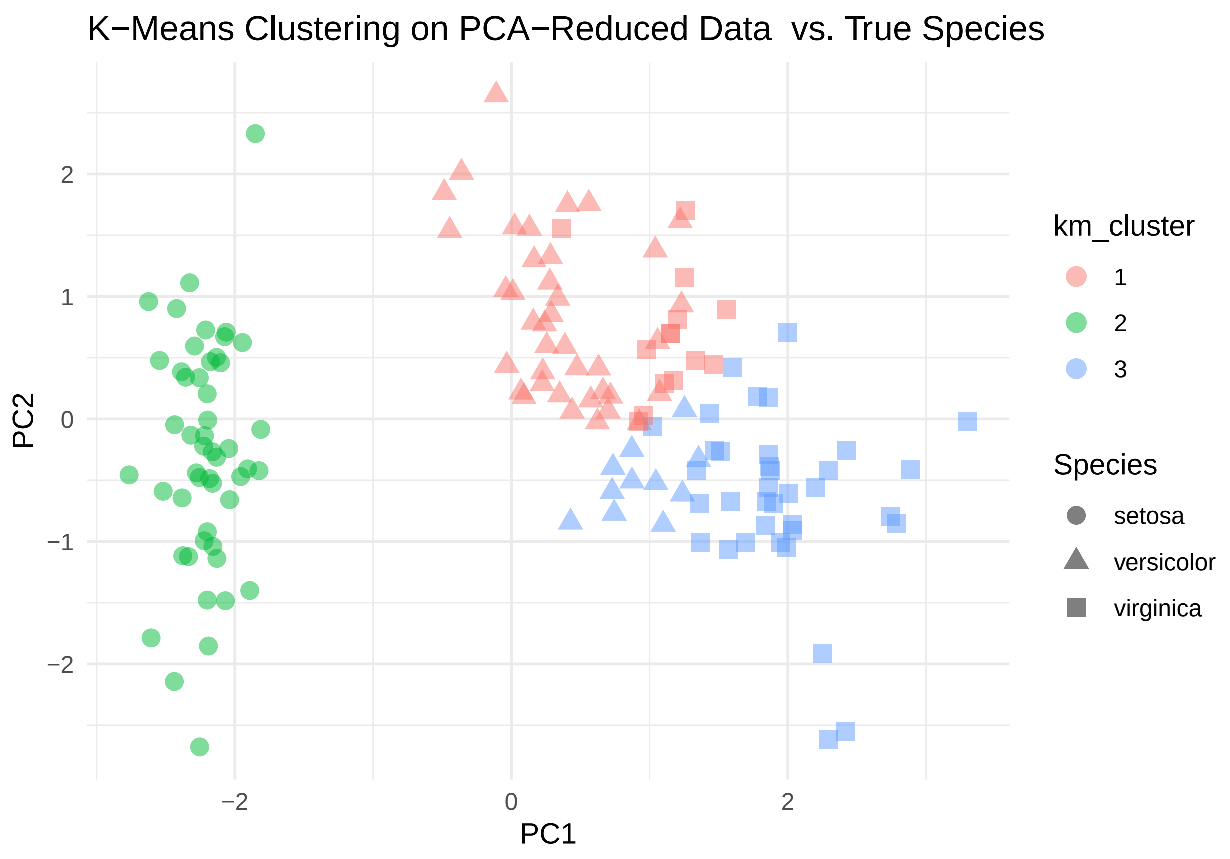
```
##      1    2    3    4    5
## 1 0.00 1.17 0.84 1.10 0.26
## 2 1.17 0.00 0.52 0.43 1.38
## 3 0.84 0.52 0.00 0.28 0.99
## 4 1.10 0.43 0.28 0.00 1.25
## 5 0.26 1.38 0.99 1.25 0.00
```

```r
# Calculate Manhattan distance, more robust to outliers
dist_matrix <- dist(iris_data, method = "manhattan")
```

```r
#----------------------------------------------------------------------
# PCA + K-Means Clustering, often better than raw features + K-Means
#----------------------------------------------------------------------


# We should use PCA before clustering for high dimensional and noisy data
# Use only first two PCs for clustering
pca_scores <- pca$x[, 1:2]

# Run kmeans on PCA reduced data
km_pca <- kmeans(pca_scores, centers = 3, nstart = 25)

# Visualized clusters
pca_df <- as.data.frame(pca_scores)
pca_df$km_cluster <- as.factor(km_pca$cluster)

ggplot(pca_df, aes(PC1, PC2, color = km_cluster)) +
  geom_point(alpha = 0.5) +
  labs(title = "K-Means Clustering on PCA-Reduced Data") +
  theme_minimal()
```



```r
# Add actual species to see how well clustering match true classes
pca_df$Species <- iris$Species

ggplot(pca_df, aes(PC1, PC2, color = km_cluster, shape = Species)) +
  geom_point(size = 3, alpha = 0.5) +
  labs(title = "K-Means Clustering on PCA-Reduced Data  vs. True Species") +
  theme_minimal()
```

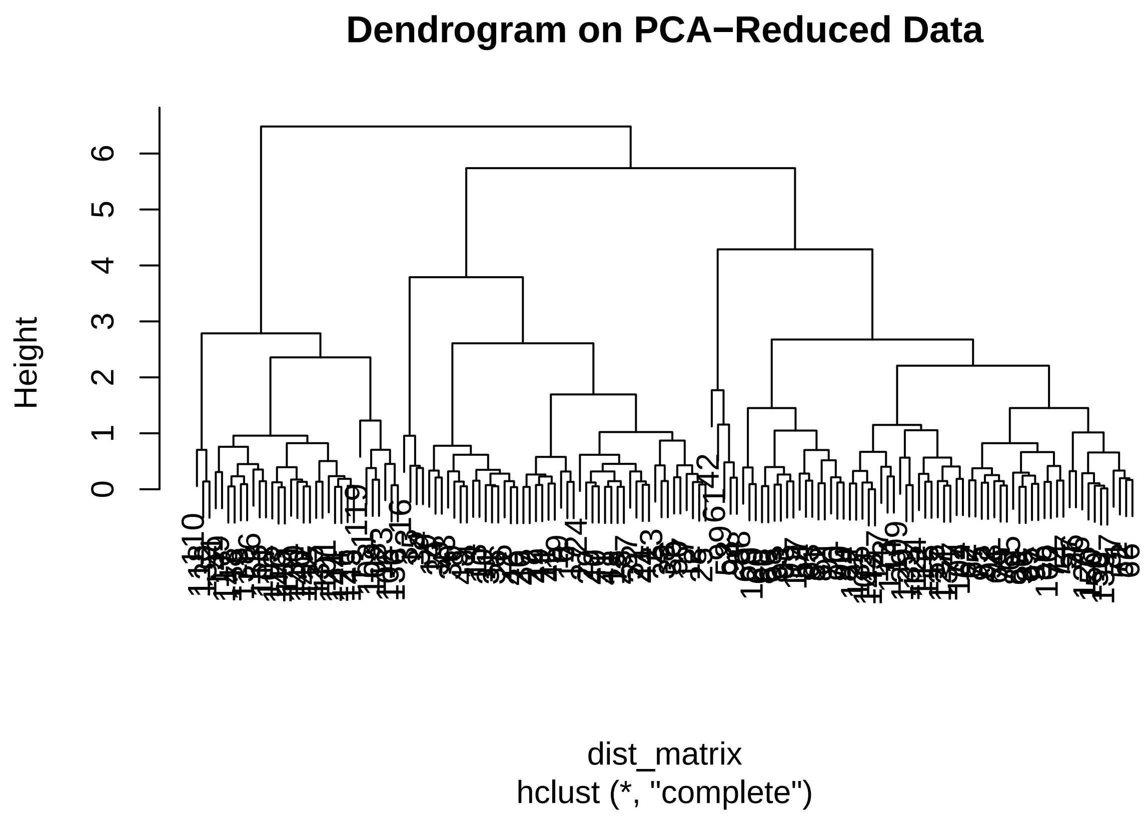


```r
# Compare table
table(pca_df$km_cluster, iris$Species)
```

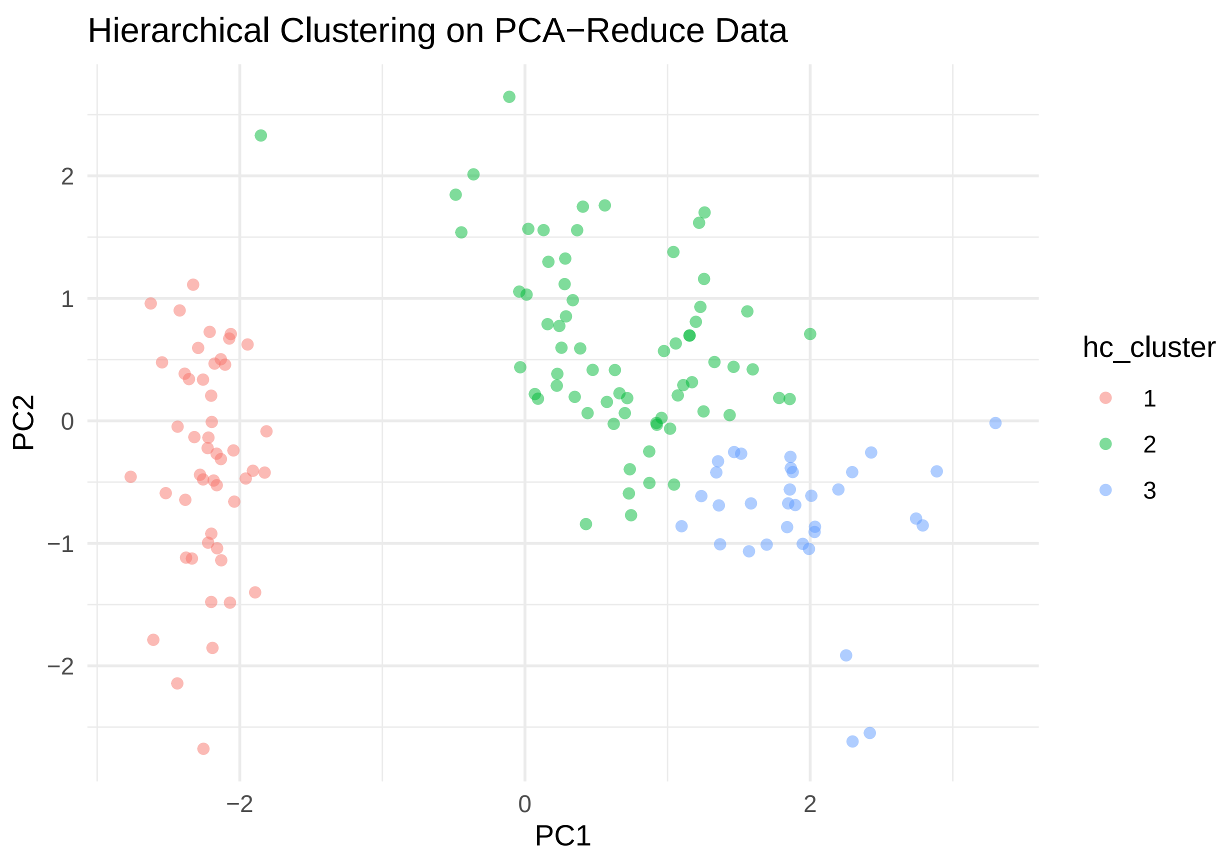
```
##
##     setosa versicolor virginica
##   1      0         39        14
##   2     50          0         0
##   3      0         11        36
```

```r
#---------------------------------
# PCA + Hierarchical Clustering
#---------------------------------


# Compute distance matrix
dist_matrix <- dist(pca_scores)

# Run hierarchical clusting on PCA reduced data
# method = "complete", uses the maximum distance between cluster points
hc_pca <- hclust(dist_matrix, method = "complete")
```

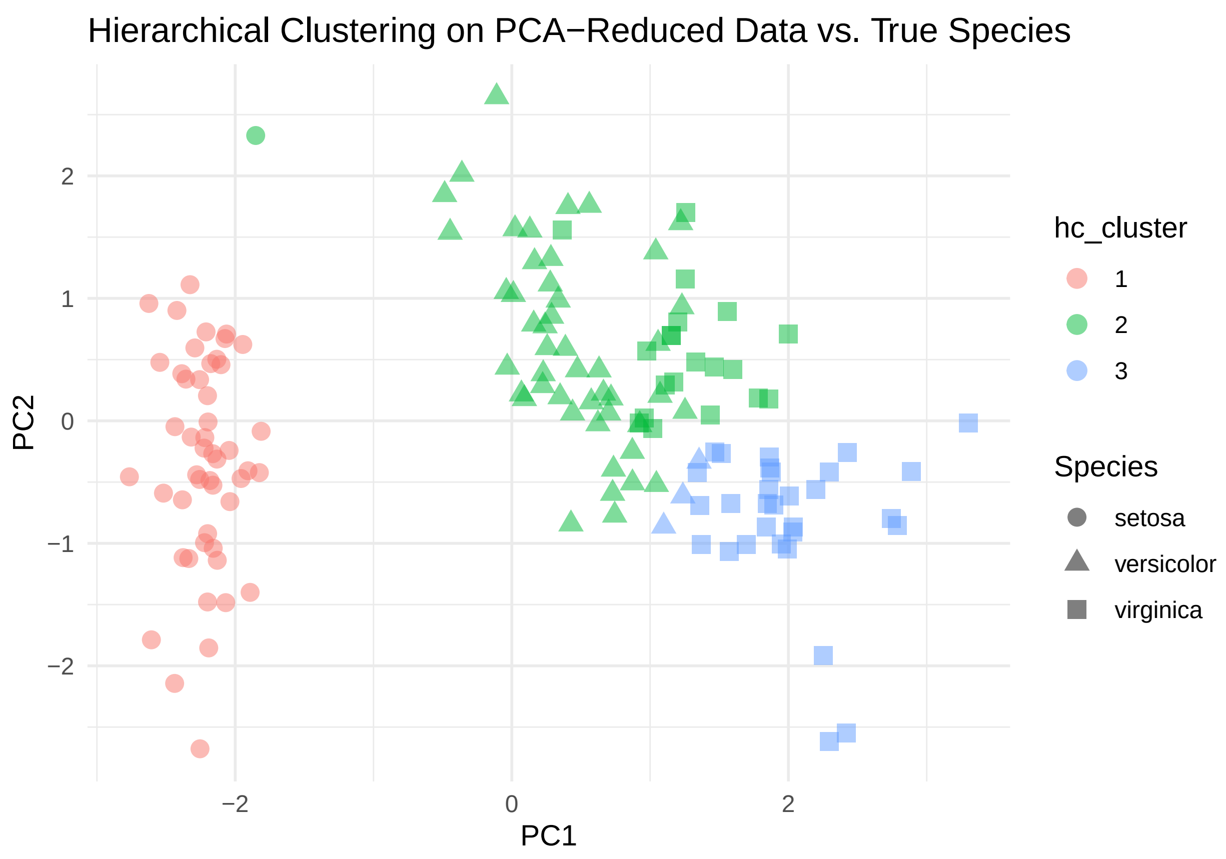
```r
# Plot dendrogram
plot(hc_pca, main = "Dendrogram on PCA-Reduced Data")
```



```r
# Cut the tree into clusters
hc_cluster <- cutree(hc_pca, k = 3)

# Add to PCA data
pca_df$hc_cluster <- as.factor(hc_cluster)

# Visualize hierarchical clusters in PCA space
ggplot(pca_df, aes(PC1, PC2, color = hc_cluster)) +
  geom_point(alpha = 0.5) +
  labs(title = "Hierarchical Clustering on PCA-Reduce Data") +
  theme_minimal()
```



```r
# Compare to actual species
ggplot(pca_df, aes(PC1, PC2, color = hc_cluster, shape = Species)) +
  geom_point(size = 3, alpha = 0.5) +
  labs(title = "Hierarchical Clustering on PCA-Reduced Data vs. True Species") +
  theme_minimal()
```



```
# Compare table
table(pca_df$hc_cluster, iris$Species)
```

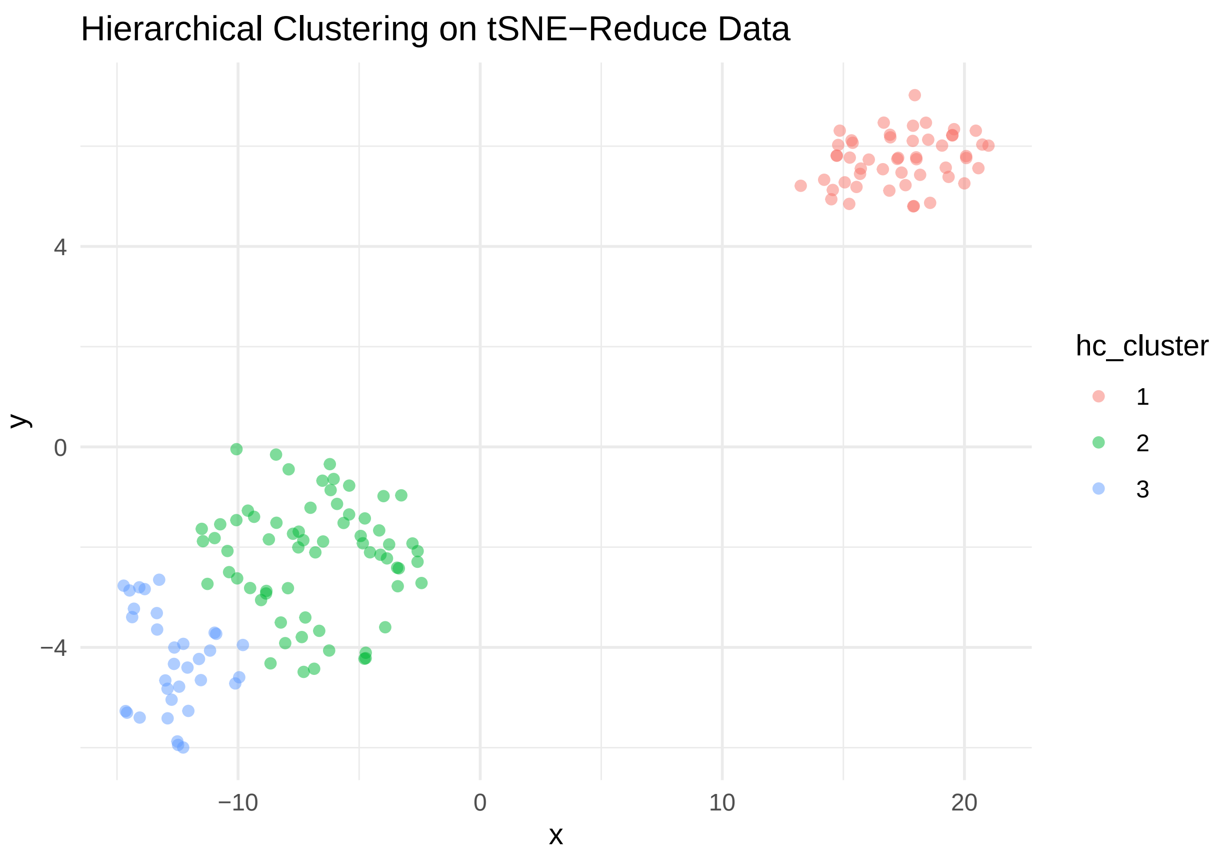
```
##
##     setosa versicolor virginica
##   1     49          0         0
##   2      1         47        20
##   3      0          3        30
```

```
#---------------------------------------
# t-SNE and how to implement it in R
#---------------------------------------



# t-SNE t-distributed Stochastic Neighbor Embedding
# t-SNE is a nonlinear dimensional reduction technique, local structure only
# t-SNE expects a numeric matrix without missing value and no duplicates
# scale data first iris_data <- scale(iris[, 1:4])

# Remove duplicates
iris_unique <- unique(iris_data)

iris_labels <- iris$Species

# Run t-SNE
```

```r
library(Rtsne)
set.seed(79)
tsne <- Rtsne(
  iris_unique, # scaled and unique
  dims = 2, # reduce to 2 dimensions
  perplexity = 30, # balances attention between local and global aspects (range 5-50)
  verbose = TRUE # show process
)
```

```
## Performing PCA
## Read the 149 x 4 data matrix successfully!
## OpenMP is working. 1 threads.
## Using no_dims = 2, perplexity = 30.000000, and theta = 0.500000
## Computing input similarities...
## Building tree...
## Done in 0.01 seconds (sparsity = 0.713752)!
## Learning embedding...
## Iteration 50: error is 46.534606 (50 iterations in 0.01 seconds)
## Iteration 100: error is 45.393362 (50 iterations in 0.01 seconds)
## Iteration 150: error is 46.225935 (50 iterations in 0.01 seconds)
## Iteration 200: error is 44.749192 (50 iterations in 0.01 seconds)
## Iteration 250: error is 44.811513 (50 iterations in 0.01 seconds)
## Iteration 300: error is 0.499532 (50 iterations in 0.01 seconds)
## Iteration 350: error is 0.298740 (50 iterations in 0.01 seconds)
## Iteration 400: error is 0.211054 (50 iterations in 0.01 seconds)
## Iteration 450: error is 0.165856 (50 iterations in 0.01 seconds)
## Iteration 500: error is 0.165387 (50 iterations in 0.01 seconds)
## Iteration 550: error is 0.163753 (50 iterations in 0.01 seconds)
## Iteration 600: error is 0.166064 (50 iterations in 0.01 seconds)
## Iteration 650: error is 0.165389 (50 iterations in 0.01 seconds)
## Iteration 700: error is 0.164275 (50 iterations in 0.01 seconds)
## Iteration 750: error is 0.163558 (50 iterations in 0.01 seconds)
## Iteration 800: error is 0.163244 (50 iterations in 0.01 seconds)
## Iteration 850: error is 0.163661 (50 iterations in 0.01 seconds)
## Iteration 900: error is 0.163867 (50 iterations in 0.01 seconds)
## Iteration 950: error is 0.163029 (50 iterations in 0.01 seconds)
## Iteration 1000: error is 0.163518 (50 iterations in 0.01 seconds)
## Fitting performed in 0.16 seconds.
```

```r
# Cluster the tsne-reduced data using hierarchical clustering
# Compute distance metrix
dist_matrix <- dist(tsne$Y)

# Perform hierarchical clustering
# ward.D2 works well with Euclidean distances for compact clusters
hc_tsne <- hclust(dist_matrix, method = "ward.D2")

# Cut the dendrogram tree into 3 clusters
hc_cluster <- cutree(hc_tsne, k = 3)

# Visualize the clusters on the t-SNE plot
tsne_df <- data.frame(
  x = tsne$Y[, 1],
  y = tsne$Y[, 2],
  hc_cluster = factor(hc_cluster)
  )

ggplot(tsne_df, aes(x, y, color = hc_cluster)) +
  geom_point(alpha = 0.5) +
  labs(title = "Hierarchical Clustering on tSNE-Reduce Data") +
  theme_minimal()
```



```r
# Compare with actual species
# Note that we unique the iris_data for Rtsne()
table(tsne_df$hc_cluster, iris$Species[!duplicated(iris_data)])
```

```
##
##     setosa versicolor virginica
##   1     50          0         0
##   2      0         50        16
##   3      0          0        33
```

```r
#--------------------------------------
# UMAP and how to implement it in R
#--------------------------------------
```

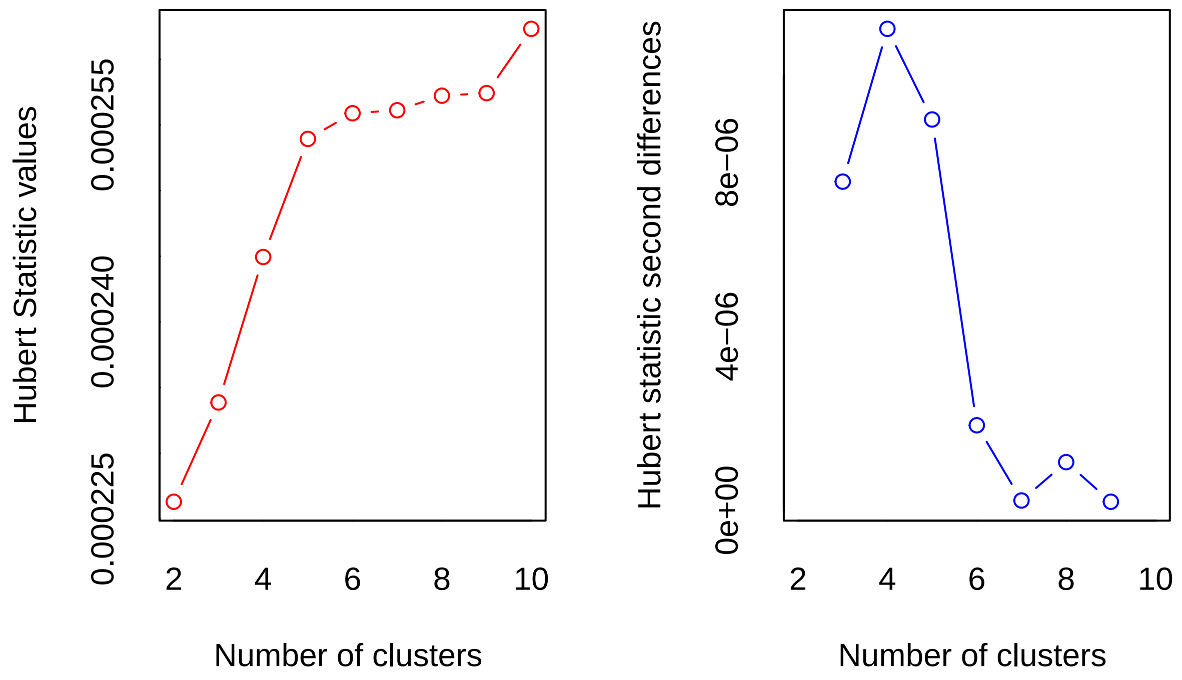
```r
# UMAP Uniform Manifold Approximation and Projection
# Balance between structure (local and global), speed, and clustering

if(!require(umap)) install.packages("umap")
```

```
## Loading required package: umap
```

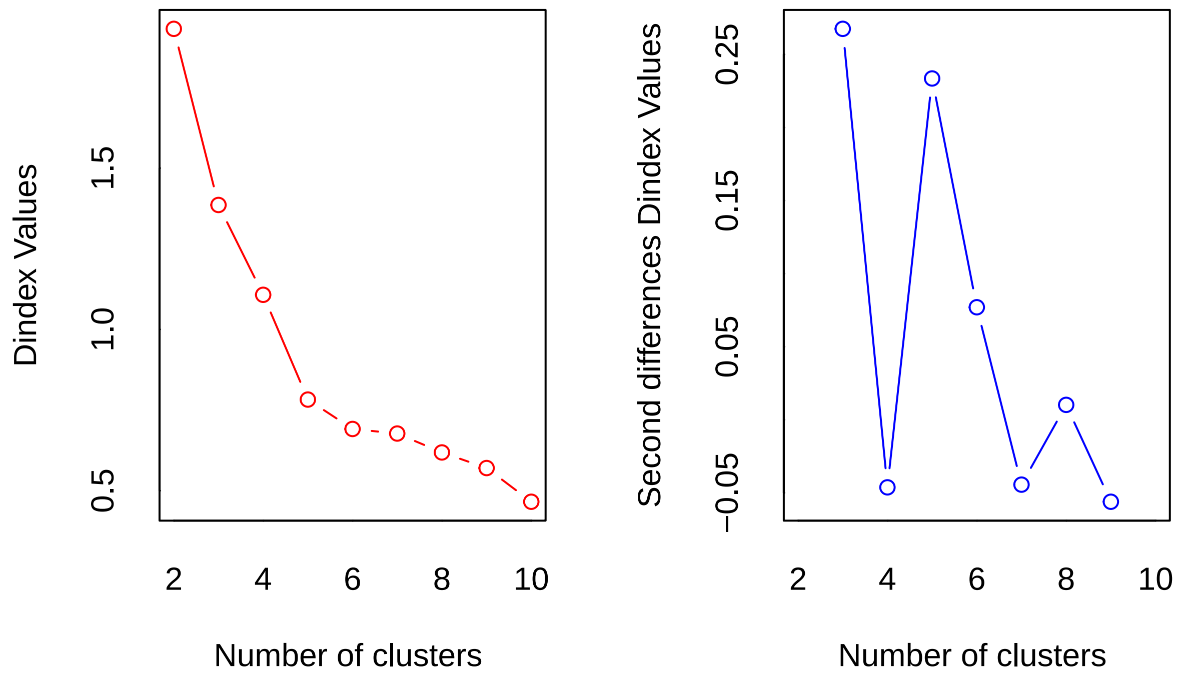
```r
library(umap)
set.seed(79)
umap <- umap(iris_data) # return a matrix of 2D coordinates (2 columns)

# Cluster the umap-reduced data using k-means clustering
# Automatically determine the optimal number of clusters using NbClust package
if(!require(NbClust)) install.packages("NbClust")
```

```
## Loading required package: NbClust
```

```r
library(NbClust)

set.seed(79)
nb <- NbClust(data = umap$layout,
              distance = "euclidean",
              min.nc = 2, # try number of clusters from 2 to 10
              max.nc = 10,
              method = "kmeans"
              )
```



```
## *** : The Hubert index is a graphical method of determining the number of clusters.
##                 In the plot of Hubert index, we seek a significant knee that corresponds to a
##                 significant increase of the value of the measure i.e the significant peak in Hubert
##                 index second differences plot.
##
```

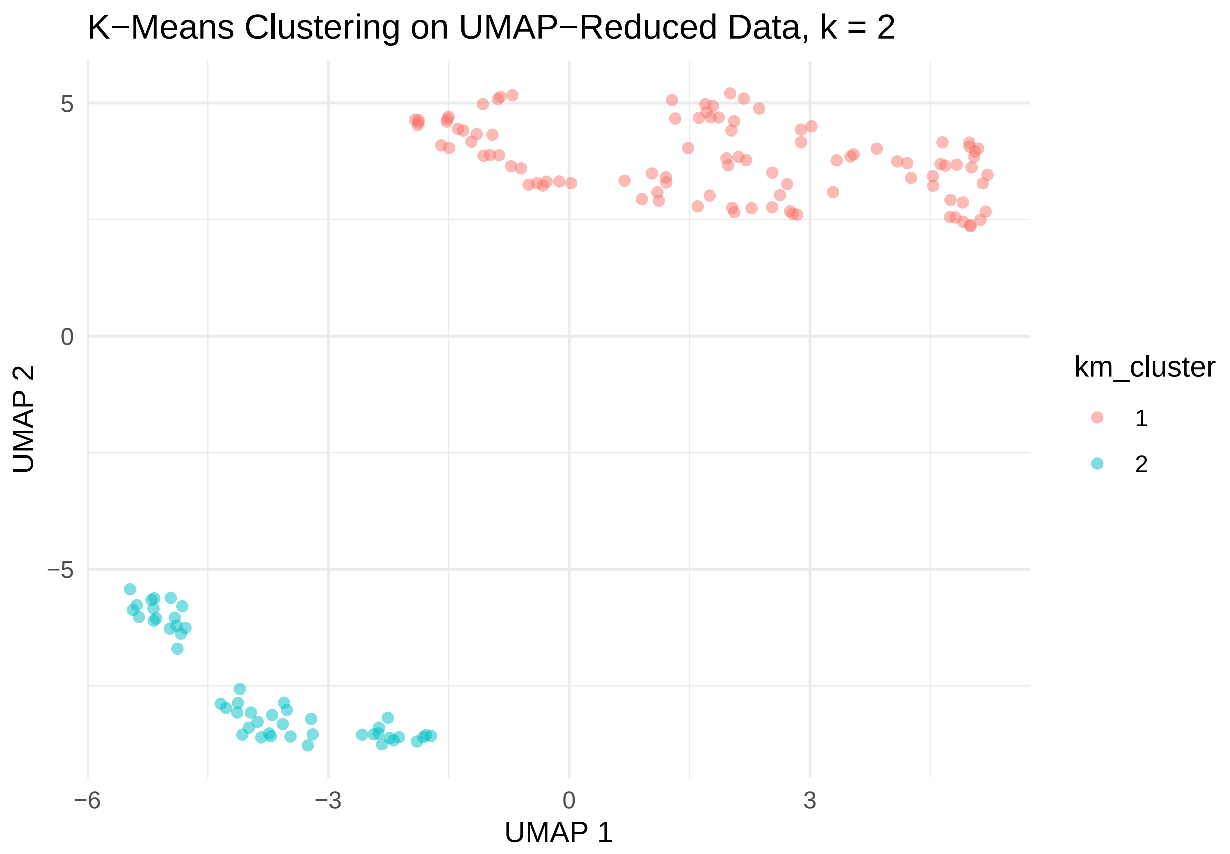


```
## *** : The D index is a graphical method of determining the number of clusters.
##                 In the plot of D index, we seek a significant knee (the significant peak in Dindex
##                 second differences plot) that corresponds to a significant increase of the value of
##                 the measure.
##
## *******************************************************************
## * Among all indices:
## * 11 proposed 2 as the best number of clusters
## * 5 proposed 3 as the best number of clusters
## * 2 proposed 5 as the best number of clusters
## * 2 proposed 6 as the best number of clusters
## * 1 proposed 9 as the best number of clusters
## * 3 proposed 10 as the best number of clusters
##
##                    ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  2
##
##
## *******************************************************************
```

```
# Check the best number of clusters
best_k <- nb$Best.nc[1] # Majority rule result
print(paste("Best number of clusters according to Nbclust:", best_k))
```

```
## [1] "Best number of clusters according to Nbclust: 2"
```

```r
# Apply k-means with optimal k
set.seed(79)
km_umap <- kmeans(umap$layout, centers = best_k, nstart = 25)
km_cluster <- km_umap$cluster

# Visualize the result
umap_df <- data.frame(
  x = umap$layout[, 1],
  y = umap$layout[, 2],
  km_cluster = factor(km_cluster)
)

ggplot(umap_df, aes(x, y, color = km_cluster)) +
  geom_point(alpha = 0.5) +
  labs(title = paste("K-Means Clustering on UMAP-Reduced Data, k =",best_k),
       x = "UMAP 1",
       y = "UMAP 2") +
  theme_minimal()
```



```r
# Compare to actual species
table(umap_df$km_cluster, iris_labels)
```

```
##    iris_labels
##     setosa versicolor virginica
##    1      0         50        50
##   2     50          0         0
```

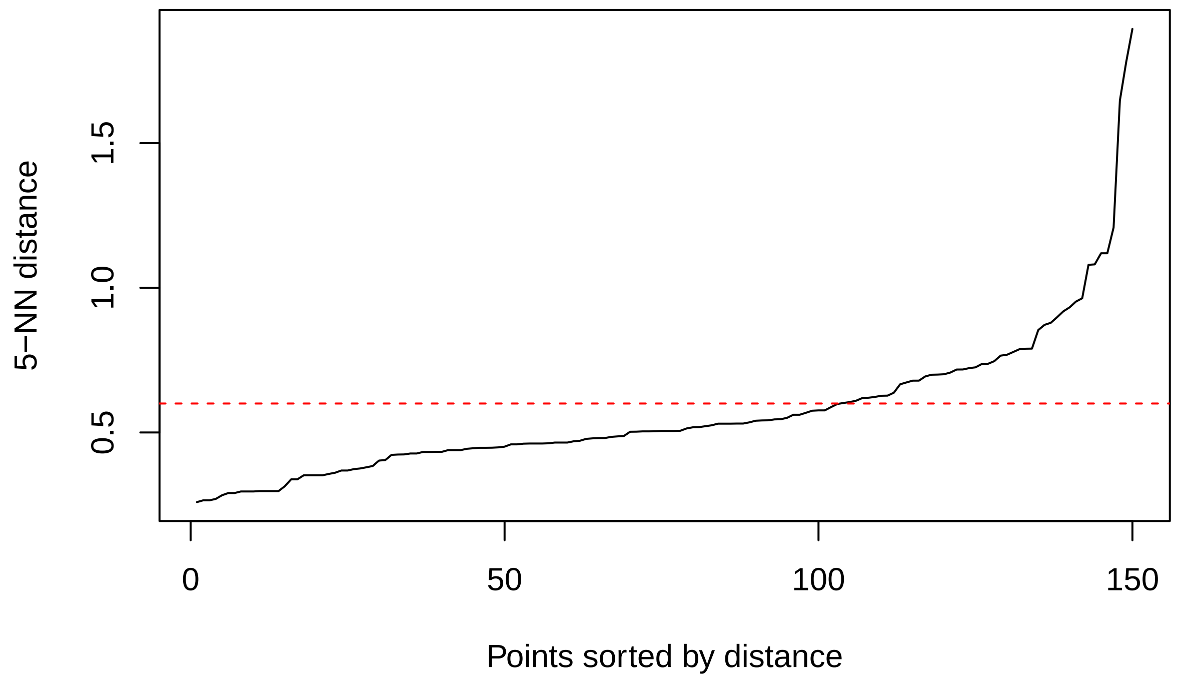
```r
#------------------------------------
# DBSCAN and how implement it in R
#------------------------------------


# DBSCAN Density-Based Spatial Clustering of Applications with Noise
# DBSCAN is a clustering algorithm that groups together points that are densely packed,
# and labels low-density points as outliers.

if(!require(dbscan)) install.packages("dbscan")
```

```
## Loading required package: dbscan

##
## Attaching package: 'dbscan'

## The following object is masked from 'package:stats':
##
##     as.dendrogram
```

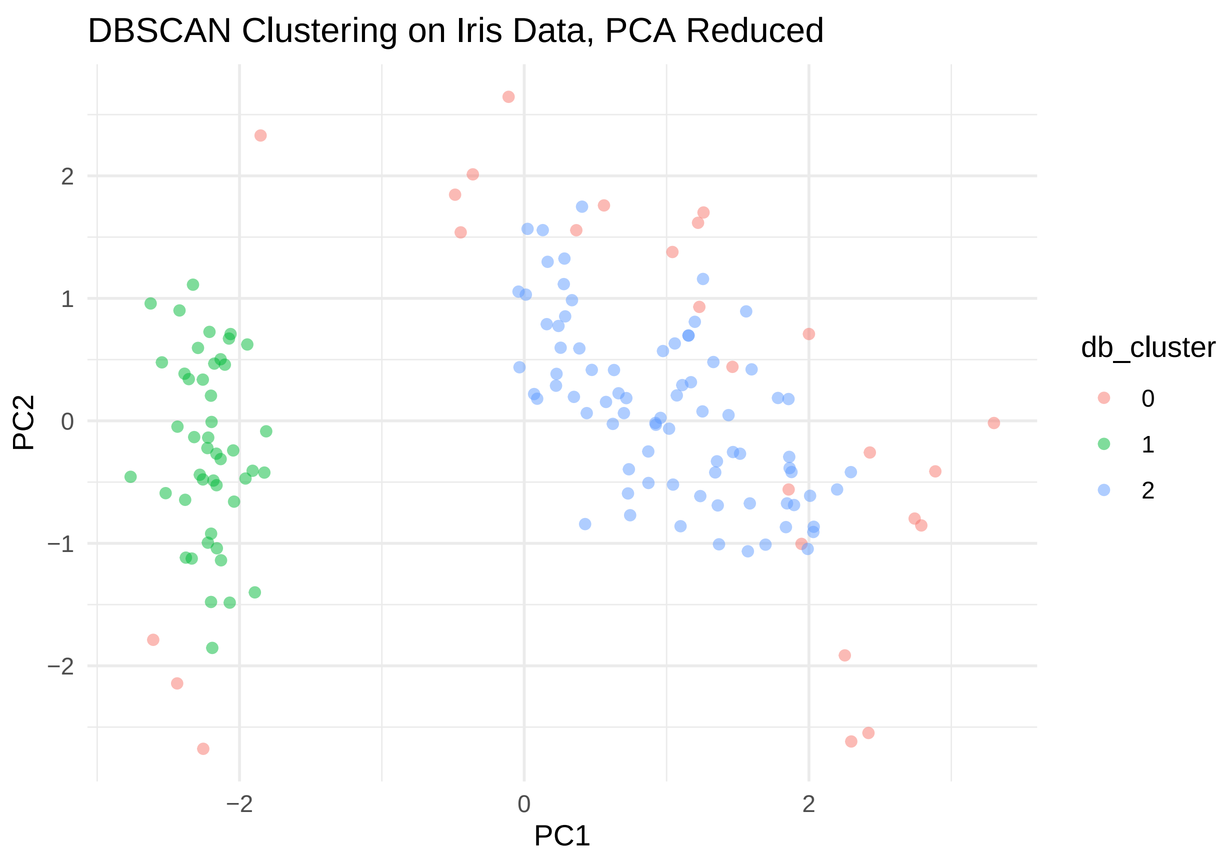
```r
library(dbscan)

# Run DBSCAN
db <- dbscan(iris_data, # scaled data
             eps = 0.5, # The radius (epsilon neighborhood) around a point to consider it a neighbor
             minPts = 5 # Minimum number of points required to form a dense region (core point)
             )

# View results
db$cluster # Cluster labels, 0 = noise
```

```
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1
##  [38] 1 1 1 1 0 1 1 1 1 1 1 1 1 2 2 2 2 2 2 0 0 2 0 0 2 0 2 2 2 2 2 0 2 2 2 0 2
##  [75] 2 2 2 2 2 2 2 2 2 2 0 2 0 2 2 2 2 2 0 2 2 2 2 0 2 0 2 2 2 2 0 0 0 0 0 2
## [112] 2 2 2 0 2 2 0 0 0 2 2 0 2 2 0 2 2 2 0 0 0 2 2 2 0 0 2 2 2 2 2 2 2 2 2 2 2
## [149] 0 2
```

```r
# Chose mintPts properly
minPts = ncol(iris_data) + 1 # or slightly higher

# Chose eps properly
# Use kNNdistplot() from dbscan package to find the "elbow"
kNNdistplot(iris_data, k = 5) # chose k = minPts
abline(h = 0.6, col = "red", lty = 2) # Example esp = 0.6
```



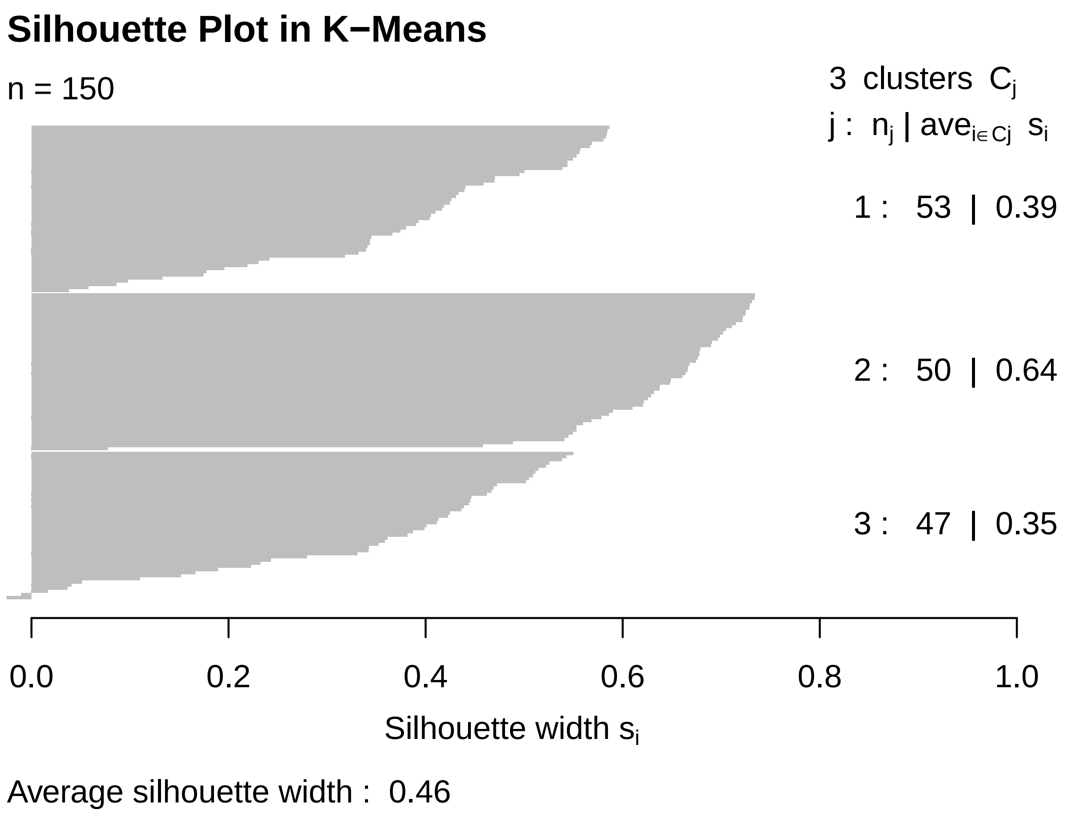
```r
# Run DBSCAN with selected minPts and eps
db_2 <- dbscan(iris_data, eps = 0.6, minPts = 5)
db_cluster <- as.factor(db_2$cluster) # 0 = noise

# visualize using first two principal components (or t-SNE or UMAP)
pca_df$db_cluster <- db_cluster

ggplot(pca_df, aes(PC1, PC2, color = db_cluster)) +
  geom_point(alpha = 0.5) +
  labs(title = "DBSCAN Clustering on Iris Data, PCA Reduced",
       x = "PC1",
       y = "PC2") +
  theme_minimal()
```



```r
#------------------------------------------------------------------------------
# Metrics are used to evaluate clustering (Silhouette score)
#------------------------------------------------------------------------------


# Measures how similar a point is to its own cluster compared to other clusters.
# ~1 = good fit within its cluster. 0 = between two clusters. Negative = wrong cluster

if(!require(cluster)) install.packages("cluster")
```

```
## Loading required package: cluster
```

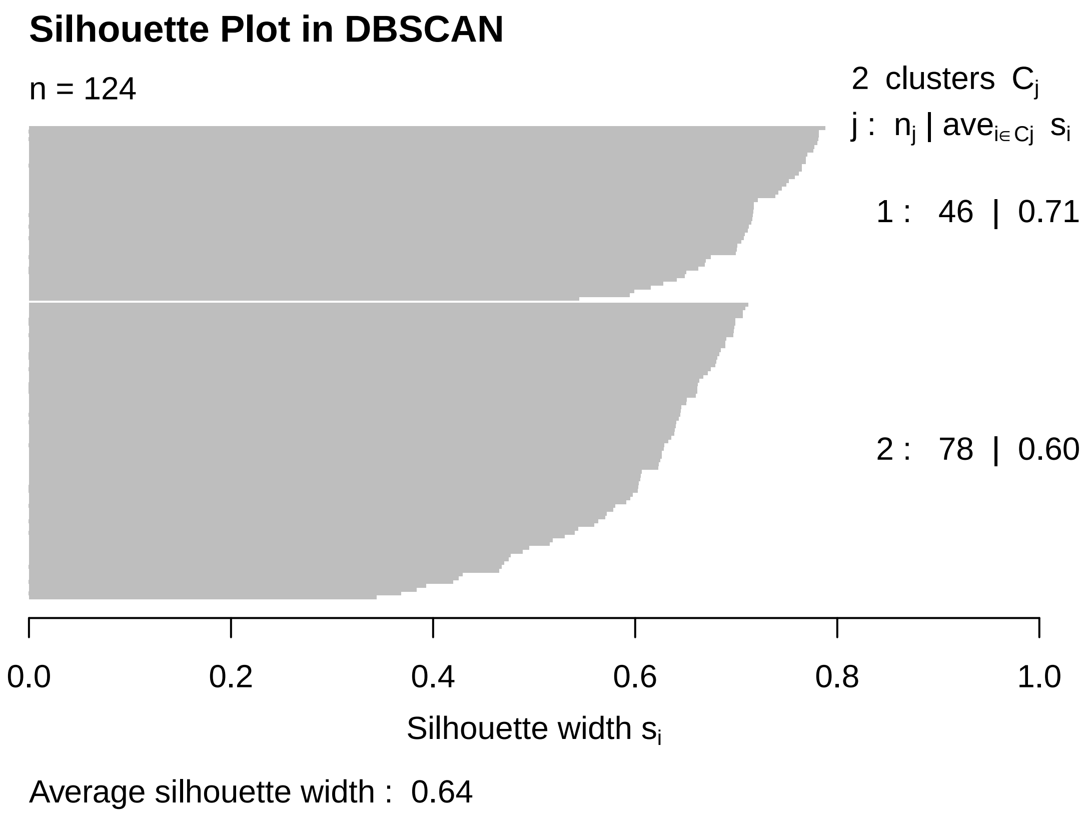
```r
library(cluster)

# In k-means clustering
km <- kmeans(iris_data, centers = 3, nstart = 25)

sil_km <- silhouette(km$cluster, dist(iris_data)) # Silhouette score

plot(sil_km, main = "Silhouette Plot in K-Means")
```



```r
mean(sil_km[, 3]) # Average silhouette width
```

```
## [1] 0.4599482
```

```r
# In DBSCAN clustering
db_2 <- dbscan(iris_data, eps = 0.6, minPts = 5)

db_cluster <- db_2$cluster
valid <- db_cluster != 0

sil_db <- silhouette(db_cluster[valid], dist(iris_data[valid, ]))

plot(sil_db, main = "Silhouette Plot in DBSCAN")
```



```r
mean(sil_db[, 3])
```

```
## [1] 0.6418947
```

In conclusion, this project illustrated how unsupervised learning can be applied to real-world datasets to reveal hidden patterns, support exploratory analysis, and even assist in automated categorization. All without requiring predefined labels. The Iris dataset served as a valuable and interpretable example, but the methods demonstrated here can scale to more complex and high-dimensional data in practical scenarios.